COMP SCI 3004/7064 Operating Systems Practical 2 – Virtual Memory Simulation Report

Group Name: PRAC 2 Group DEC

Students:

Domenic Lombardo - a1756807

Ebubekir Pulat - a1833363

Cuang William - a1843487

September 17, 2023

INTRODUCTION

To effectively make use of main memory available to a program, regularly accessed pages can be kept in memory, with the rest residing in a disk drive. The methodology of deciding which pages should be kept in main memory is critical, as unfavourable selections can result in high page fault rates, which can heavily decrease overall computing performance.

Page replacement algorithms apply policies which determine the pages to be replaced when main memory is full, and a page from disk is needing to be stored in that memory. Such algorithms include first-in-first-out (FIFO), random (rand), least-recently used (LRU) and the clock algorithm. FIFO algorithms replace the page in memory which was inserted there first, and random algorithms, likewise to their name, randomly pick a page to replace. LRU algorithms replace the page, which was accessed the longest ago, and the clock algorithm replaces pages it reasonably estimates to be the least recently used.

This report intends to look into the effectiveness of the above-mentioned page replacement algorithms, including how they fare with specific application traces. These traces contain portions of page access history of real programs. Evaluation of the algorithms aims to be performed through running them over these traces, recording the resultant page fault rates as well as the number of disk reads and writes, and then comparing the results. Additionally, the nature of the application traces will be analysed, which can help deduce the applicability of the page replacement algorithms for certain memory access patterns.

METHODS

The page replacement algorithms to be tested are the FIFO, rand, LRU and clock algorithms. These algorithms will be tested with each of the application traces which include the gcc, bzip, sixpack, and swim traces; each of these traces contain 1 million memory references. Each algorithm, for each trace, will be tested with main memory sizes (in terms of page number) of 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096 and 8192. All tests will be performed in quiet mode, which provides a concise summary of the page replacement algorithms' performances, for further analysis.

The experiments are performed in a directory containing the application traces, and the program (memsim.exe) that simulates the functionality of the page replacement algorithms. Using the terminal, the experiments are executed using command lines of the form: ./memsim {gcc, bzip, sixpack, swim}.trace {main memory size} {rand, fifo, lru, clock} quiet

The outcomes recorded for each test include the page fault rate of the algorithm, the number of reads on disk executed, as well as the number of writes to disk performed.

RESULTS

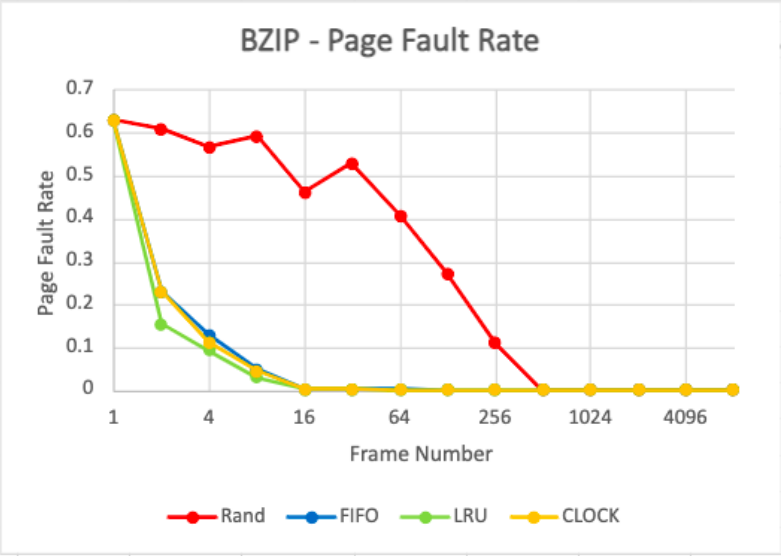

Figure 1: The page fault rates recorded by the simulator on the 'bzip' trace

Bzip:

From Figure 1, it is observable that the LRU, FIFO and clock algorithms initially have sharp falls in page fault rate as main memory slightly largens, which precedes a less drastic, more linear decrease, then finally a plateau hovering over a 0% page fault rate.

The page fault rates of the FIFO and clock algorithms decrease from approximately 63% to around 23% by simply increasing the main memory size from 1 page to 2 pages. However, the LRU algorithm outperforms them, dropping to a basement-dwelling 15.44% (Appendix A). Evidently, page fault rates quickly fall as the number of frames in main memory increases, which suggests that the trace has a high degree of temporal and/or spatial locality. Thus, extremely low main memory space is needed to hold a majority of the ‘popular’ pages.

The bzip trace's locality is further emphasised by its minor number of unique memory accesses, as the most optimal page fault rate is at an extremely low 0.03% (Appendix A). This 0.03% corresponds to 317 unique disk accesses, out of 1 million accesses, which translates to heavy traffic to an immensely select group of physical frames (Appendix B).

Finally, another indicator of the trace’s locality is the random algorithm’s considerably worse page fault performance compared to the other algorithms, which better account for access history. Specifically, while the random algorithm still showcases page fault rates of above 40%, the other algorithms have already dropped under 10%. Moreover, only until 512 pages are in main memory, does the random algorithm finally close-in on a perfect page fault rate, whereas its competitors have already done so with a mere 16 pages. Therefore, only 16 pages in memory are needed in most cases to efficiently operate on the 'bzip' trace, or 512 for absolute confidence no matter the page replacement policy used.

Although the memory usage in bzip is highly central, Figure 1 surprisingly showcases minimal benefit to utilising the clock algorithm over FIFO, even though the clock method is much more 'history respecting'. On the other hand, LRU has a visibly superior graph line, discernibly moving below the rest which

suggests that the clock algorithm's approximation of LRU's functionality is inadequate to take advantage of the bzip trace's locality.

GCC:

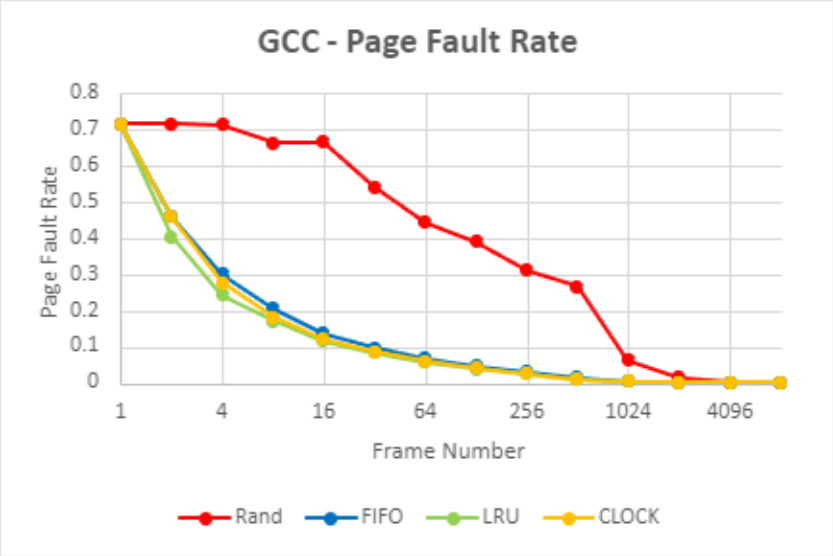

Figure 2: The page fault rates recorded by the simulator on the gcc trace

In Figure 2, the algorithms in the gcc trace reduce page faults more gradually than in the bzip trace, starting at 71.61% and plateauing at 0.29% at frame number 4096 (the optimal frame size) (Appendix C). This is higher than in the bzip trace, which has a low optimal page fault rate of 0.03% at only 512 frames. This implies that more additional memory is needed to cope with the gcc trace's more unique memory accesses. However, if around 1% is taken as an acceptable fault rate, the gcc trace will only require 2048 frames (half of the optimal frame size).

Again, the 'random' method performs worse on page faults. Even at 256 frames, it still pagefaults at 31.09%. Whereas for LRU, when the frame number is raised just from 1 to 2, it performs a satisfying decrease in page faults to 40.4% (31.21% decrease, the highest compared to other algorithms) (Appendix C). This indicates LRU's excellent ability to capture the gcc trace's temporal and/or spatial locality. FIFO and clock algorithms decline much less, hovering around 46% within the same frame interval (1-2), making LRU the best performing algorithm with low frame numbers (Appendix C).

The clock algorithm's more advanced LRU approximation doesn't appear to outperform FIFO in this trace, which raises issues about its usefulness. Though, unique locality properties of the gcc trace may explain it. The trace's temporal and/or spatial locality may not work well with clock's second-chance mechanism, making it less effective. Furthermore, regarding the trace's access patterns, if they are inconsistent or unpredictable, the clock approach's LRU approximation may not capture this complexity, making it function more like FIFO than expected.

Examining the gcc trace over time shows that all algorithms' page fault rates optimise at around 4096 frames, which implies that the trace has varied memory access pattern phases and that the algorithms adjust differently. For instance, the LRU algorithm's quick drop in page fault rates early on may imply it is preferable for the volatile gcc trace. This then suggests that LRU may indeed be the go-to algorithm for

smaller memories, but its performance diminishes over time as frame sizes increase. In contrast, clock and FIFO algorithms are more stable but less adaptable, making them less successful initially but more dependable over time. Regardless, if the choice comes down to LRU, clock, or FIFO, LRU still exhibits good performance in all scenarios in this particular trace.

Sixpack:

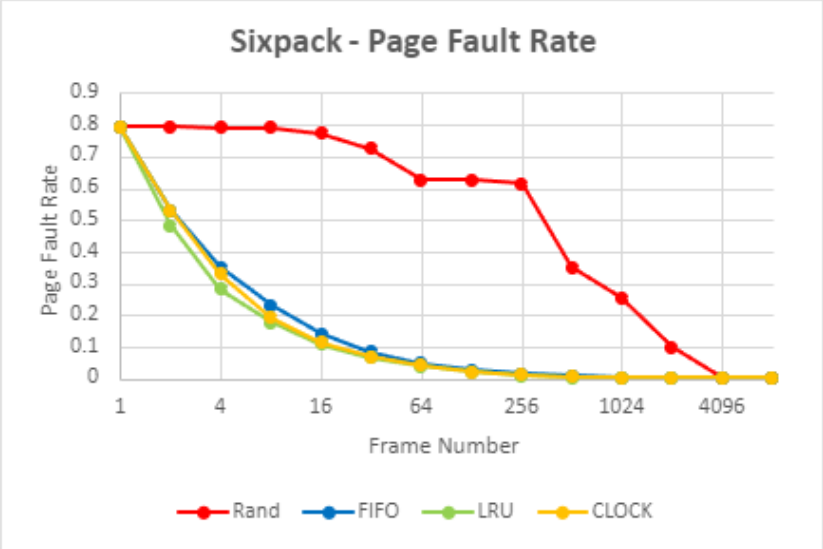

Figure 3: The page fault rates recorded by the simulator on the sixpack trace

Visible in Figure 3, is a somewhat logarithmic shape formed by the page fault rates of the LRU, FIFO and clock algorithms, as main memory increases in size. These algorithms fall from page fault rates of roughly 80% with 1 page in memory, to 10-15% with 16 pages, and near 0% with 512 pages, and plateaus from then on. Each instance the frame number in main memory doubles, the page fault rate seemingly halves correspondingly.

The same is not seen with the random algorithm, which showcases a graph line almost opposite to the others. The random algorithm operated horrendously on the sixpack trace, as it required 512 pages, to simply move under a 60% page fault rate. Further, 4096 pages were needed to drop clear of 10%; 4096 exceeds the number of unique page accesses in the sixpack trace, 3890, emphasising the failure of the random policy (Appendix D). The lack-luster results of the random algorithm may come down to the sheer number of unique pages being accessed in the trace, reducing the likelihood of 'random' success, from unlikely to extremely improbable. Another reason could be that the random seed utilised to implement the 'randomness' of the algorithm, was limited in variance, leading to the algorithm repeatedly choosing the unpopular pages, in a cyclic pattern.

Poor performance from the random algorithm strongly suggests the sixpack trace possesses some level of temporal and/or spatial locality. However, the distinct separation of the page fault rates from the LRU, FIFO and clock algorithms, with LRU the best, and FIFO the poorest, more than suggests that the memory access behaviour in the trace is repetitive, and often. Repetition may involve accessing the exact same address regularly, or constantly accessing neighbouring addresses in a single physical frame.

Although, the degree of repetition is relatively moderate, as 512 pages were needed to achieve reasonable page fault rates.

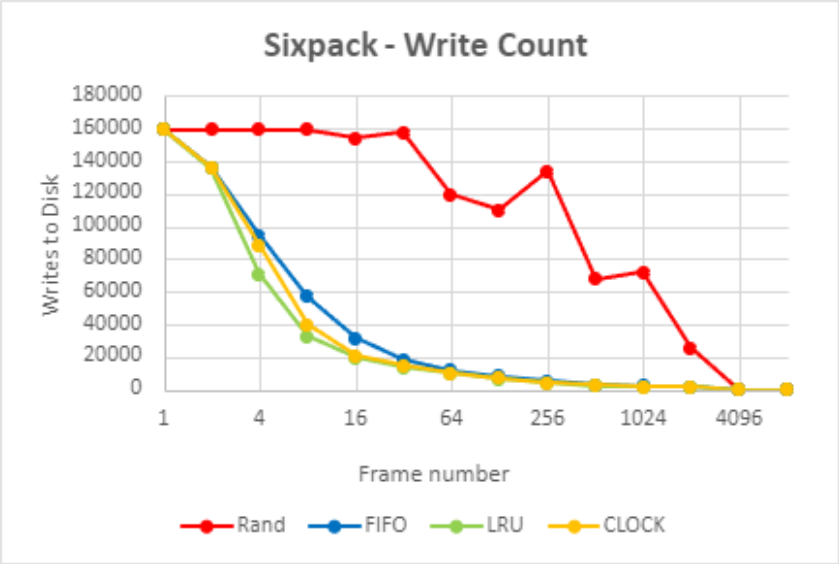

Figure 4: The number of disk writes recorded by the simulator on the sixpack trace

On the other hand, with the number of disk writes shown in Figure 4, the superiority of the LRU and clock algorithms over FIFO is much more obvious, especially at memory sizes of 8 and 16. This level of isolation between the algorithms is not replicated in Figure 3. This isolation indicates that in the sixpack trace, general accessing and reading is more sporadic, but when it comes to writing, conducting changes in memory, it is concentrated to fewer physical frames.

Swim:

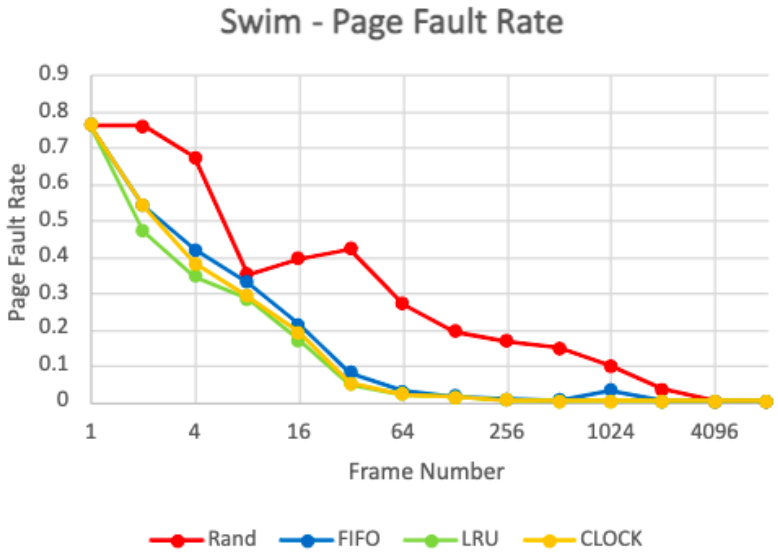

Figure 5: The page fault rates recorded by the simulator on the swim trace

In Figure 5, it is visible that the FIFO, LRU and clock algorithms resemble the similar logarithmic shape seen in the sixpack, gcc and bzip traces. However, unlike the previously discussed traces, the reduction in page faults is more gradual and linear, rather than logarithmic. For example, the FIFO, LRU and clock algorithms in the bzip trace reach page fault rates of 10% and below, at 4 to 8 frames, while in swim, these page fault numbers are only reached at 32 frames. From this, it can be assumed that the swim trace accesses memory with less locality than the other traces. Additionally, while the random algorithm performs poorly in all four traces, in swim, its page fault rates are more closely aligned to those of FIFO, LRU and clock. This further emphasises that when compared to the other traces, locality is less present in the swim trace, and therefore highlights that the replacement algorithms have less of an impact on the performance of the program. However, further investigation would have to be conducted to determine if the success of the random algorithm in this trace was not by chance, and can be reproduced over multiple instances.

When comparing the different algorithms, it's notable that when frame numbers are smaller, LRU performed between 2% to 10% better than clock, and 13% to 40% better than FIFO which resulted in a page fault decrease of up to 7% (Appendix E). However, as the frame number reaches 64 and the page fault rates move below 3%, the differences between these three algorithms become negligible. Mirroring the trends seen in previously discussed traces, the random algorithm reaches negligible page fault values only when the frame numbers are much higher. In the case of swim, the random algorithm reaches below 3% page fault values at 2048 frames.

CONCLUSION

From the experiments performed, and the resultant data recorded, it is clear that the LRU algorithm was most effective for page replacement with the four application traces. Consistently, the LRU algorithm showcased page fault rates below all of its competitors, on the other hand, the random algorithm consistently fell far behind. The random algorithm's performance was so poor, it appears to be an outlier of failure, rather than a contestant to be best page replacement algorithm. Reasoning behind the random algorithm's disappointing results, may be due to the relatively large number of unique pages being accessed in the traces, or the random seed (in the simulator's code) used was insufficient to provide an adequately 'random' nature. In future practicals, a large number of experiments should have been performed with the random algorithm, and with varying random seeds, to reduce random error and have an improved showcase of the random algorithm's performance.

On the other hand, the clock and FIFO algorithms fought it out in-between, usually performing quite similarly, but LRU's imitator (clock) slightly edged out to be second-best. Despite the differences between the algorithms, they all achieve perfect page fault rates, on all traces, when there are 4096 pages in memory.

Regarding the traces, the bzip trace discernibly displayed the highest access locality, as only few main memory frames, 16, were needed to repeatedly receive page hits. Aside from that anomaly, the other traces generally showcased more moderate locality, translating to more logarithmic graph lines, rather than downward exponential as seen in bzip. The logarithmic shapes translated to page fault rate plateaus (nearing perfect page fault performance) at higher page numbers, specifically around 256 to 512 for swim and sixpack, and 512 to 1024 for gcc. The swim trace seemingly displayed the strangest results, with the random algorithm performing far less poorly than seen in other traces, perhaps coming down to 'random' success, and not swim's incredible lack of locality.

APPENDIX

Appendix A: Table showing raw data from the bzip trace, for the page faults rates of the random, FIFO, LRU and clock algorithms on varying frame numbers

Frame Number	Random	FIFO	LRU	Clock
1	0.6297	0.6297	0.6297	0.6297
2	0.6081	0.2288	0.1544	0.2288
4	0.5657	0.1286	0.0928	0.1114
8	0.5924	0.0478	0.0307	0.0462
16	0.4619	0.0038	0.0033	0.0035
32	0.5285	0.0025	0.0021	0.0022
64	0.4069	0.0015	0.0013	0.0013
128	0.273	0.0009	0.0008	0.0008
256	0.1099	0.0005	0.0004	0.0004
512	0.0003	0.0003	0.0003	0.0003
1024	0.0003	0.0003	0.0003	0.0003
2048	0.0003	0.0003	0.0003	0.0003
4096	0.0003	0.0003	0.0003	0.0003
8192	0.0003	0.0003	0.0003	0.0003

Appendix B: Table showing raw data from the bzip trace, for the number of disk reads of the random, FIFO, LRU and clock algorithms on varying frame numbers

Frame Number	Random	FIFO	LRU	Clock
1	629737	629737	629737	629737
2	608069	228838	154429	228838
4	565696	128601	92770	111442
8	592427	47828	30691	46164
16	461933	3820	3344	3468
32	528531	2497	2133	2203
64	406852	1467	1264	1318
128	273032	891	771	802
256	109898	511	397	432

512	317	317	317	317
1024	317	317	317	317
2048	317	317	317	317
4096	317	317	317	317
8192	317	317	317	317

Appendix C: Table showing raw data from the gcc trace, for the page faults rates of the random, FIFO, LRU and clock algorithms on varying frame numbers

Frame Number	Random	FIFO	LRU	Clock
1	0.7161	0.7161	0.7161	0.7161
2	0.7156	0.4609	0.404	0.4609
4	0.7137	0.3029	0.2438	0.277
8	0.6625	0.2054	0.1712	0.1819
16	0.6661	0.1385	0.1166	0.1217
32	0.5431	0.0981	0.0844	0.0877
64	0.444	0.0703	0.0591	0.0616
128	0.3922	0.0485	0.0408	0.043
256	0.3109	0.0317	0.0253	0.0268
512	0.266	0.0156	0.0104	0.0115
1024	0.0632	0.0067	0.0044	0.0048
2048	0.0166	0.0034	0.0029	0.0029
4096	0.0029	0.0029	0.0029	0.0029
8192	0.0029	0.0029	0.0029	0.0029

Appendix D: Table showing raw data from the sixpack trace, for the number of disk reads of the random, FIFO, LRU and clock algorithms on varying frame numbers

Frame Number	Random	FIFO	LRU	Clock
1	792379	792379	792379	792379
2	792310	529237	483161	529237
4	791209	351810	282620	329368

8	790371	230168	176496	191760
16	773134	140083	108682	115206
32	726034	85283	67747	70849
64	624583	48301	41186	41949
128	626836	27778	21090	23229
256	616029	15440	11240	11876
512	350887	8089	5823	6119
1024	256298	5492	4468	4618
2048	99483	4314	3951	3993
4096	3890	3890	3890	3890
8192	3890	3890	3890	3890

Appendix E: Table showing raw data from the swim trace, for the page fault rates of the random, FIFO, LRU and clock algorithms on varying frame numbers

Frame Number	Random	FIFO	LRU	Clock
1	0.7634	0.7634	0.7634	0.7634
2	0.7604	0.5415	0.4707	0.5415
4	0.6734	0.4195	0.3469	0.3807
8	0.3525	0.3309	0.2854	0.2935
16	0.395	0.2143	0.172	0.1918
32	0.4233	0.0816	0.0483	0.053
64	0.2704	0.0304	0.0217	0.0226
128	0.1961	0.0175	0.0132	0.0153
256	0.1685	0.0086	0.0057	0.006
512	0.1504	0.0049	0.0036	0.0038
1024	0.1003	0.033	0.0028	0.0029
2048	0.0364	0.0028	0.0026	0.0026
4096	0.0025	0.0025	0.0025	0.0025
8192	0.0025	0.0025	0.0025	0.0025